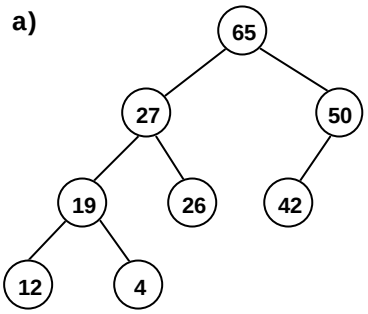
Trabajo Práctico Nº 10

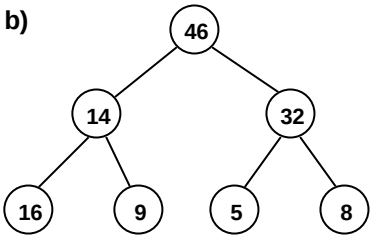
Tema: Montículos - Arboles Binarios de Expresión – Colas de Prioridad.

1. Decir si los siguientes árboles son completos, llenos, montículos o ninguno de los anteriores:

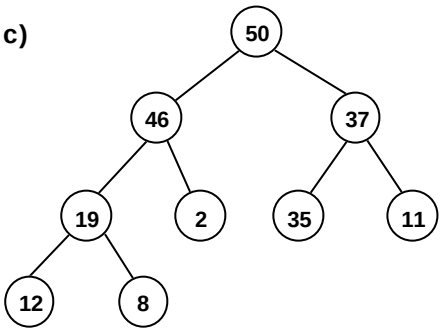
a)



b)



c)

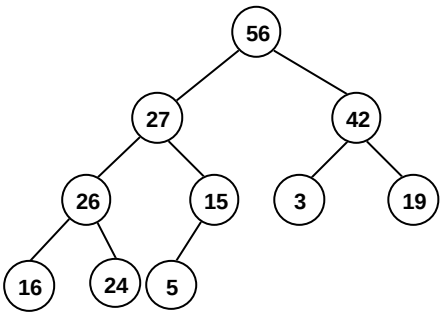


2. Una cola de prioridad que contiene caracteres se implementa como un montículo, la precondición afirma que esta cola de prioridad no puede contener elementos duplicados. ¿Qué valores pueden almacenarse en las posiciones 8 a 10 del array, de forma que las prioridades de un montículo se satisfagan?

Montículo

Z	F	J	E	B	G	H	?	?	?
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]

3. Una cola de prioridad se implementa como un montículo:



Mostrar como quedaría el montículo anterior después de esta serie de operaciones:

```
ColaDePrioridad CP = new ColaDePrioridad();
int X, Y, Z;

CP.insertar(28);
```

Trabajo Práctico Nº 10

```
CP.insertar(2);
CP.insertar(40);
X = CP.suprimir();
Y = CP.suprimir();
Z = CP.suprimir();
```

b. ¿Cuáles serian los valores de X, Y y Z después de la serie de operaciones del inciso a?

4. Agregar a la clase Montículo los recorridos inorden y postorden.

5. Escribir los métodos limpiarColaPrioridad() y estaLlena(), usando la implementación de montículo.

6. Una cola de Prioridad se implementa como una lista enlazada. Escribir la clase para esta implementación.(insertar, suprimir, etc.)
Comparar con las operaciones con las correspondientes a la implementación de montículo. ¿Qué implementación es mejor/peor para cada operación?

Nota: ordenar la lista de mayor a menor.

7. Una Cola de prioridad se implementa como un árbol binario de búsqueda. Escribir la clase para esta implementación.(insertar, suprimir, etc.)
Comparar con las operaciones con las correspondientes a la implementación de montículo. ¿Qué implementación es mejor/peor para cada operación?

8. Una cola de Prioridad se implementa como un montículo, está ordenada para acceder al elemento más pequeño antes que al elemento mayor. ¿Cómo necesitarían ser modificados los algoritmos en la Cola de prioridad? ¿Cómo necesitarían ser modificados los algoritmos en las operaciones sobre montículos? ¿Se puede implementar un montículo en una lista enlazada?

9. Realizar los siguientes ejercicios utilizando una cola de prioridad implementada en un montículo.

Los usuarios del sistema tienen prioridades relativas de acuerdo con su numero de ID:

Usuarios	0 – 99	mayor (por ejemplo, ejecutivos de la empresa)
Usuarios	100 – 199	siguiente más alta (por ejemplo, secretarios ejecutivos)
Usuarios	200 – 299	siguiente más alta (por ejemplo, jefes de grupo)
--		
--		
--		
Usuarios	800 – 899	anterior a la más baja (por ejemplo, programadores)
Usuarios	900 – 999	más baja (sus trabajos sólo se ejecutaran a las 3 de la madrugada los fines de semana)

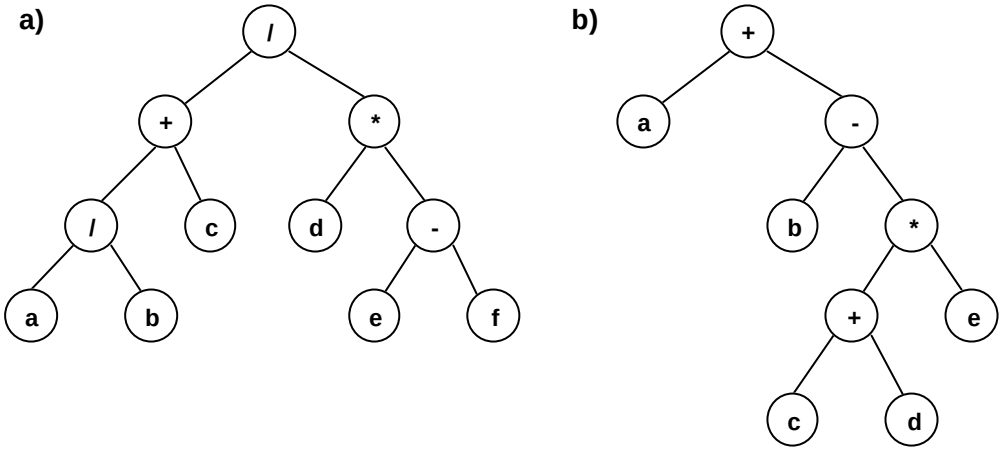
a. Escribir un método **añadirTrabajo** que reciba un numero de **id** de usuario y un **token** representando el trabajo que hay que ejecutar y los añada a la cola.
b. Escribir un método **obtenerSigTrabajo** que devuelva el **token** del trabajo para ejecutar.

Hay que apagar el sistema para el mantenimiento. Todos los trabajos que estén esperando para ejecutarse se quitaran de la cola. Sin embargo, este es un sistema amistoso que notifica a los usuarios

10. Escribir un programa que evalúe una expresión aritmética. La expresión puede ingresarse en la notación que se crea mas adecuada.

11. Mostrar la notación en preorden, inorden y postorden de los siguientes árboles binarios de expresión (Recordar que debe utilizar paréntesis para la notación en inorden).

Trabajo Práctico Nº 10



12. Evaluar el árbol binario de expresión del ejercicio 11.a con los siguientes valores para las variables a - f:

a = 6; b = 2; c = 5; d = 2; e = 8; f = 7

13. Evaluar el árbol binario de expresión del ejercicio 11.b con los siguientes valores para las variables a - e:

a = 5; b = 12; c = 2; d = 3; e = 2

14. Dibujar el árbol binario de una expresión que representa la siguiente expresión en preorden:

+ * a b / + c d e

15. Dibujar el árbol binario de una expresión que representa la siguiente expresión en inorden:

(((a - b) + c) * ((d + e) / f)) - g

16. Dibujar el árbol binario de una expresión que representa la siguiente expresión en postorden:

a b + c / d *

17. Escribir un procedimiento recursivo para imprimir la representación en postorden de un árbol binario de expresión. Usar la implementación con registro variante de los nodos del árbol.

18. La construcción de un árbol binario de expresiones puede realizarse utilizando una expresión en notación prefija, como se dictó en la asignatura, mediante el método obtenerSimbolo(). Pero también puede construirse utilizando una expresión en notación postfija. Por supuesto que el algoritmo será diferente. A continuación se describe y especifica el método para realizar la construcción del árbol a partir de una expresión en notación postfija. (La notación postfija coincide con un recorrido postorden de un árbol binario de expresión). El formato básico de una expresión postfija es:

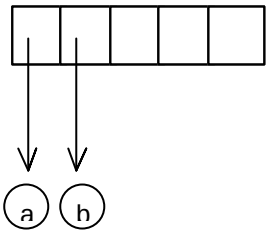
Operando 1 Operando 2 Operador.

Leemos una expresión, símbolo por símbolo, mediante el método obtenerSimbolo(). Sí el símbolo es un operando, creamos un árbol de un nodo y metemos un puntero a él en una pila. Si el símbolo es un operador, sacamos los punteros a dos árboles A1 y A2 de la pila (primero A1) y se forma un nuevo árbol cuya raíz es el operador, con dos hijos izquierdo y derecho apuntados por A2 y A1, respectivamente. Un puntero a este nuevo árbol se mete en la pila.

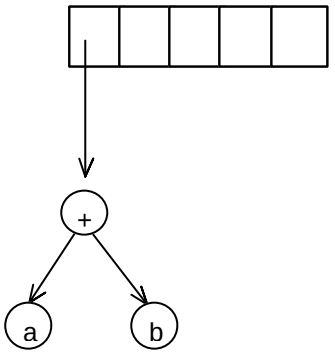
Ejemplo: a b + c d e + * * < = > (a + b) * c * (d + e)

Trabajo Práctico Nº 10

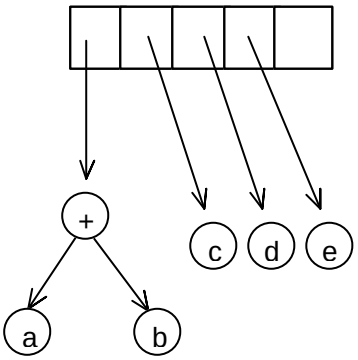
Los primeros dos símbolos (a y b) son operandos, así que creamos Nodos y metemos punteros a ellos en una pila.



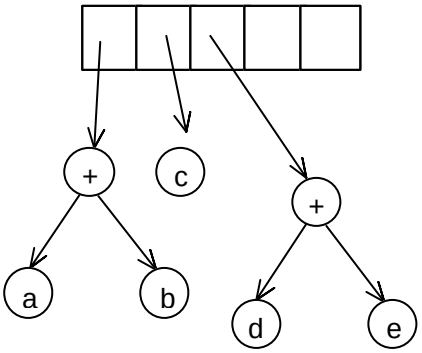
Después, se lee un "+", así que se sacan dos apuntadores a Nodos de la pila, se forma un árbol nuevo y se mete a la pila un puntero a éste.



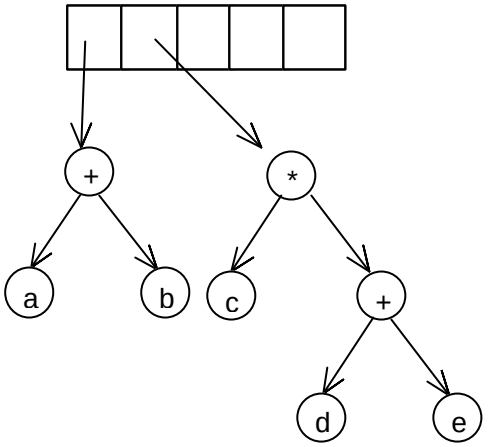
A continuación, se leen c, d y e, y para cada uno se forma un Nodo y los puntero correspondientes se meten en la pila.



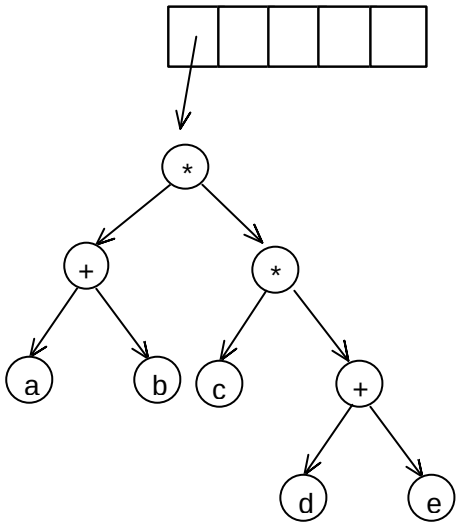
Ahora se lee un "+", lo cual hace que se combinen dos árboles.



Trabajo Práctico Nº 10



Continuando se lee un "*", así que se sacan dos punteros y se forma un árbol nuevo con "*" como raíz.
Para terminar, se lee el último símbolo, se combinan dos árboles y se deja en la pila un puntero al árbol final.



Una vez creado el árbol de expresión el algoritmo para evaluar el mismo, no cambia. Es decir, se puede usar el método EVAL.

Se pide:

a. Escriba el algoritmo para crear el árbol a partir de una expresión en notación postfija, utilizando el método descripto.
b. Implemente el algoritmo en un programa, que además permita recorrer el árbol en postorden y preorden, y evalué la expresión ingresada.